T3DBarGraph Manual

Delphi FireMonkey 3D Bar Charts

Overview

T3DBarGraph

T3DBarGraph is a Delphi / FireMonkey 3D bar chart component built on top of `TViewport3D`.

It supports dynamic data, positive and negative values, labels, transparent planes, mouse interaction, bar selection, and a mesh-based rendering mode for larger datasets.

Download PDF manual

View on GitHub

Main Features

- Delphi / FireMonkey 3D bar chart component.
- Dynamic bar data with row and column labels.
- Positive and negative values.
- Configurable colors for bars, selected bars, grids, planes, labels, and legend.
- Configurable plane transparency through `PlaneOpacity`.
- Mouse and keyboard navigation.
- Bar selection with a floating 3D legend.
- Mesh-based rendering path for larger datasets.
- Batched loading with `BeginDataUpdate` and `EndDataUpdate`.

Current Release

The current public release is `v0.1.0`.

Prebuilt test executables, when available, are published as GitHub Release assets.

When To Use It

T3DBarGraph is useful when you want a Delphi-native 3D visualization for dataset-style information, especially when rows, columns, and values have a natural visual relationship.

For very dense datasets, the performance test application is the best place to evaluate whether the 3D representation remains readable for your use case.

Getting Started

Getting Started

Requirements

• Delphi with FireMonkey support.

• A platform supported by FireMonkey 3D.

The project was created as a Delphi FireMonkey component. Depending on your Delphi version, RAD Studio may update project metadata when opening the project files.

Repository Contents

The repository includes:

• `U3DBarGraph.pas`: main component source code.

• `T3DBarGraphPackage.dproj`: installable package project.

• `T3DBarGraphDemo.dproj`: demo application.

• `Test.dproj`: performance test application.

Install The Component

• Open `T3DBarGraphPackage.dproj` or `T3DBarGraphPackage.dpk` in Delphi.

• Build the package.

• Install the package into the IDE.

• The component should appear in the `UofW` component palette as `TBarGraph`.

Run The Demo

• Open `T3DBarGraphDemo.dproj`.

• Build and run the project.

• Use the demo controls to change visual settings, scale behavior, camera state, and lighting.

Run The Performance Test

• Open `Test.dproj`.

• Build and run the project.

• Use the `1k`, `5k`, `10k`, and `50k` buttons to compare loading and interaction at different dataset sizes.

The performance test is useful when changing rendering, picking, transparency, camera movement, or bulk data loading.

Basic Usage

Basic Usage

Add `U3DBarGraph` to your form unit and place a `TBarGraph` on the form, either through the IDE palette or by creating it in code.

```
uses
U3DBarGraph, System.UIConsts;
procedure TForm1.FormCreate(Sender: TObject);
begin
BarGraph1.BeginDataUpdate;
try
BarGraph1.XLabel := 'Season';
BarGraph1.YLabel := 'Period';
BarGraph1.ZLabel := 'Mean temperature';
BarGraph1.PlaneOpacity := 0.5;
BarGraph1.AddYLabel(0, '1987-1996');
BarGraph1.AddYLabel(1, '1937-1946');
BarGraph1.AddYLabel(2, '1887-1896');
BarGraph1.AddXLabel(0, 'Spring');
BarGraph1.AddXLabel(1, 'Summer');
BarGraph1.AddXLabel(2, 'Autumn');
BarGraph1.AddXLabel(3, 'Winter');
BarGraph1.Add(0, 0, -15, claGreen);
BarGraph1.Add(1, 0, 14, claPurple);
BarGraph1.Add(2, 0, 14, claRed);
BarGraph1.Add(0, 1, 25, claGreen);
BarGraph1.Add(1, 1, 25, claPurple);
BarGraph1.Add(2, 1, 25, claRed);
finally
BarGraph1.EndDataUpdate;
end;
end;
```

Updating Existing Bars

`Add(row, col, value, color)` behaves as an upsert operation.

If a bar already exists at the same row and column, the existing bar is updated instead of creating a duplicate.

```
BarGraph1.Add(0, 0, 10, claBlue);
BarGraph1.Add(0, 0, 20, claRed); // updates the same bar
```

Bulk Loading

Wrap bulk updates in `BeginDataUpdate` and `EndDataUpdate`.

```
BarGraph1.BeginDataUpdate;
try
for I := 0 to Count - 1 do
BarGraph1.Add(Row, Col, Value, Color);
finally
BarGraph1.EndDataUpdate;
end;
```

This avoids recalculating layout and repainting after every single bar.

API Reference

API Reference

This page lists the main public API exposed by `TBarGraph`.

Data Methods

```
procedure Add(row, col: Integer; Value: Single; cl: TAlphaColor = 0);
procedure AddYLabel(row: Integer; val: String);
procedure AddXLabel(col: Integer; val: String);
procedure BeginDataUpdate;
procedure EndDataUpdate;
procedure Reset;
```

Add

Adds or updates a bar at a row/column position.

```
BarGraph1.Add(0, 0, 42, claBlue);
```

Calling `Add` again with the same row and column updates the existing bar.

AddYLabel / AddXLabel

Adds or updates row and column labels.

```
BarGraph1.AddYLabel(0, 'Row A');
BarGraph1.AddXLabel(0, 'Column A');
```

BeginDataUpdate / EndDataUpdate

Use these methods around bulk data changes.

```
BarGraph1.BeginDataUpdate;
try
// Add labels and bars here.
finally
BarGraph1.EndDataUpdate;
end;
```

Reset

Restores the view state.

Axis And Scale Properties

```
property ZLabel: String;
property YLabel: String;
property XLabel: String;
property AutoScale: Boolean;
property NumTicks: Integer;
property ZMin: Single;
property ZMax: Single;
```

Visual Properties

```
property BackgroundColor: TAlphaColor;
property BarColor: TAlphaColor;
property BarSelectedColor: TAlphaColor;
```

```
property GridColor: TAlphaColor;
property XYPlaneColor: TAlphaColor;
property XZandYZPlaneColor: TAlphaColor;
property PlaneOpacity: Single;
property FontColor: TAlphaColor;
property LegendFontColor: TAlphaColor;
property LegendBackgroundColor: TAlphaColor;
```

`PlaneOpacity` is clamped from `0` to `1`; the default is `0.5`.

View Helpers

```
procedure ViewNegativePlane;
procedure ViewPositivePlane;
procedure TurnLights(Val: Boolean);
```

Use `TurnLights(False)` if you need flatter colors or want to inspect the geometry without scene lighting.

Interaction

Interaction

T3DBarGraph supports mouse and keyboard navigation.

Mouse

• Left-drag to rotate the graph.

• Hold `Ctrl` and left-drag to pan the view.

• Use the mouse wheel to zoom toward the current cursor position.

• Click a bar to select it and show its floating 3D legend.

• Click empty space to clear the current selection.

• Right-click to open the component popup menu.

Keyboard

• Arrow keys pan the view.

• `R` resets the view.

• `Home` resets the view.

Selection

Bar selection works in both rendering paths:

• normal cube rendering;

• mesh rendering for larger datasets.

In mesh mode, the component uses screen-space picking so selection remains available even when many bars are rendered through grouped meshes.

Practical Notes

For dense charts, navigation quality matters as much as raw rendering speed. If users need to inspect specific values, consider limiting the visible dataset or grouping data before rendering.

Performance

Performance

T3DBarGraph includes a performance test application in `Test.dproj`.

The test app includes buttons for:

- `1k`
- `5k`
- `10k`
- `50k`

Bulk Loading

Use `BeginDataUpdate` and `EndDataUpdate` when adding many bars.

```
BarGraph1.BeginDataUpdate;
try
for I := 0 to Count - 1 do
BarGraph1.Add(Row, Col, Value, Color);
finally
BarGraph1.EndDataUpdate;
end;
```

Mesh Rendering

For larger datasets, the component can render bars through a mesh-based path instead of creating one FireMonkey 3D object per bar.

This reduces object count and keeps interaction usable with much larger bar counts. The included performance test has been used locally with 50,000 bars.

Readability

Rendering 50,000 bars does not automatically mean the result is visually useful.

For very dense datasets, consider:

- filtering visible rows or columns;
- grouping categories;
- showing a top-N view;
- using multiple charts;
- allowing users to drill into a smaller subset.

The goal is not only to draw many bars; it is to keep the data interpretable.

Releases

Releases

Published versions are available from the GitHub Releases page.

Open GitHub Releases

Release Assets

Releases may include a prebuilt performance-test ZIP.

Use the ZIP if you want to try the test application without compiling the project in Delphi.

Generated binaries and distribution archives are not committed to the source tree. They belong in GitHub Release assets.

Source Code

Each GitHub Release includes source archives generated by GitHub for the associated tag.

For active development, clone the repository and use `main`.

Roadmap

Roadmap

The public roadmap is tracked in two places:

- `ROADMAP.md` in the repository.
- GitHub issues labeled `enhancement`.

Open roadmap issues

v0.2.0 Focus

- CSV and dataset loading helpers.
- Camera presets.
- `FitToData`.
- Better tooltip placement.
- More examples for dynamic updates.

v0.3.0 Focus

- Image export helpers.

• Optional value-based color palettes.
• Better defaults for labels, lighting, and dense charts.
• Continued refactoring of large implementation areas.

Contributing

Contributing

Contributions are welcome through GitHub issues and pull requests.

Read CONTRIBUTING.md

Useful Bug Reports

For rendering or interaction issues, include:
• Delphi/RAD Studio version.
• Target platform.
• Steps to reproduce.
• Expected behavior.
• Actual behavior.
• Screenshot or short video.

Useful Feature Requests

For feature requests, include:
• the data visualization use case;
• expected API shape, if relevant;
• sample data or screenshots;
• performance constraints.

Pull Request Notes

Before opening a pull request:
• keep the change focused;
• avoid committing generated build output;
• test visually in the demo when changing rendering, picking, labels, transparency, or navigation;
• run the performance app when changing data loading or render behavior.